TECHNISCHE UNIVERSITÄT DRESDEN

DATABASE THEORY

Lecture 10: Conjunctive Query Optimisation

Sebastian Rudolph

Computational Logic Group

Slides based on Material of Markus Krötzsch and David Carral

TU Dresden, 17th May 2021

Review

There are many well-defined static optimisation tasks that are independent of the database
$\leadsto$ query equivalence, containment, emptiness

Unfortunately, all of them are undecidable for FO queries
$\leadsto$ Slogan: "all interesting questions about FO queries are undecidable"

$\leadsto$ Let's look at simpler query languages

Optimisation for Conjunctive Queries

Optimisation is simpler for conjunctive queries

Example 10.1: Conjunctive query containment:

$$Q_1: \quad \exists x, y, z.\ R(x, y) \wedge R(y, y) \wedge R(y, z)$$

$$Q_2: \quad \exists u, v, w, t.\ R(u, v) \wedge R(v, w) \wedge R(w, t)$$

Q_1 find R-paths of length two with a loop in the middle
Q_2 find R-paths of length three

$\leadsto$ in a loop one can find paths of any length
$\leadsto$ $Q_1 \sqsubseteq Q_2$

Deciding Conjunctive Query Containment

Consider conjunctive queries $Q_1[x_1, \ldots, x_n]$ and $Q_2[y_1, \ldots, y_n]$.

Definition 10.2: A query homomorphism from Q_2 to Q_1 is a mapping μ from terms (constants or variables) in Q_2 to terms in Q_1 such that:

- μ does not change constants, i.e., $\mu(c) = c$ for every constant c
- $x_i = \mu(y_i)$ for each $i = 1, \ldots, n$
- if Q_2 has a query atom $R(t_1, \ldots, t_m)$
  then Q_1 has a query atom $R(\mu(t_1), \ldots, \mu(t_m))$

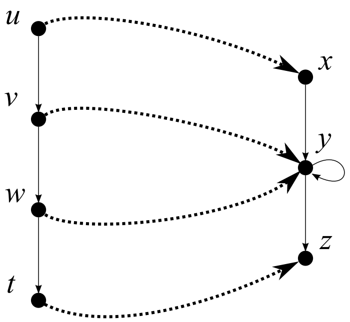
Theorem 10.3 (Homomorphism Theorem): $Q_1 \sqsubseteq Q_2$ if and only if there is a query homomorphism $Q_2 \to Q_1$.

$\leadsto$ decidable (only need to check finitely many mappings from Q_2 to Q_1)

Example

$$Q_1 : \quad \exists x, y, z.\ R(x, y) \land R(y, y) \land R(y, z)$$

$$Q_2 : \quad \exists u, v, w, t.\ R(u, v) \land R(v, w) \land R(w, t)$$



Review: CQs and Homomorphisms

If $\langle d_1, \ldots, d_n \rangle$ is a result of $Q_1[x_1, \ldots, x_n]$ over database $\mathcal{I}$ then:

- there is a mapping ν from variables in Q_1 to the domain of $\mathcal{I}$
- $d_i = \nu(x_i)$ for all $i = 1, \ldots, m$
- for all atoms $R(t_1, \ldots, t_m)$ of Q_1, we find $\langle \nu(t_1), \ldots, \nu(t_m) \rangle \in R^{\mathcal{I}}$
  (where we take $\nu(c)$ to mean c for constants c)

$\leadsto$ $\mathcal{I} \models Q_1[d_1, \ldots, d_n]$ if there is such a homomorphism ν from Q_1 to $\mathcal{I}$

(Note: this is a slightly different formulation from the "homomorphism problem" discussed in a previous lecture, since we keep constants in queries here)

Proof of the Homomorphism Theorem

"$\Leftarrow$": $Q_1 \sqsubseteq Q_2$ if there is a query homomorphism $Q_2 \to Q_1$.

(1) Let $\langle d_1, \ldots, d_n \rangle$ be a result of $Q_1[x_1, \ldots, x_n]$ over database $\mathcal{I}$.

(2) Then there is a homomorphism ν from Q_1 to $\mathcal{I}$.

(3) By assumption, there is a query homomorphism $\mu : Q_2 \to Q_1$.

(4) But then the composition $\nu \circ \mu$, which maps each term t to $\nu(\mu(t))$, is a homomorphism from Q_2 to $\mathcal{I}$.

(5) Hence $\langle \nu(\mu(y_1)), \ldots, \nu(\mu(y_n)) \rangle$ is a result of $Q_2[y_1, \ldots, y_n]$ over $\mathcal{I}$.

(6) Since $\nu(\mu(y_i)) = \nu(x_i) = d_i$, we find that $\langle d_1, \ldots, d_n \rangle$ is a result of $Q_2[y_1, \ldots, y_n]$ over $\mathcal{I}$.

Since this holds for all results $\langle d_1, \ldots, d_n \rangle$ of Q_1, we have $Q_1 \sqsubseteq Q_2$.

(See board for a sketch showing how we compose homomorphisms here)

Proof of the Homomorphism Theorem

"$\Rightarrow$": there is a query homomorphism $Q_2 \to Q_1$ if $Q_1 \sqsubseteq Q_2$.

(1) Turn $Q_1[x_1, \ldots, x_n]$ into a database $\mathcal{I}_1$ in the natural way:
   - The domain of $\mathcal{I}_1$ are the terms in Q_1
   - For every relation R, we have $\langle t_1, \ldots, t_m \rangle \in R^{\mathcal{I}_1}$ exactly if $R(t_1, \ldots, t_m)$ is an atom in Q_1

(2) Then Q_1 has a result $\langle x_1, \ldots, x_n \rangle$ over $\mathcal{I}_1$
(the identity mapping is a homomorphism – actually even an isomorphism)

(3) Therefore, since $Q_1 \sqsubseteq Q_2$, $\langle x_1, \ldots, x_n \rangle$ is also a result of Q_2 over $\mathcal{I}_1$

(4) Hence there is a homomorphism ν from Q_2 to $\mathcal{I}_1$

(5) This homomorphism ν is also a query homomorphism $Q_2 \to Q_1$.

Implications of the Homomorphism Theorem

The proof has highlighted another useful fact:

The following two are equivalent:

- Finding a homomorphism from Q_2 to Q_1
- Finding a query result for Q_2 over $\mathcal{I}_1$

$\leadsto$ all complexity results for CQ query answering apply

Theorem 10.4: Deciding if $Q_1 \sqsubseteq Q_2$ is NP-complete.

If Q_2 is a tree query (or of bounded treewidth, or of bounded hypertree width) then deciding if $Q_1 \sqsubseteq Q_2$ is polynomial (in fact LOGCFL-complete).

Note that even in the NP-complete case the problem size is rather small (only queries, no databases)

Application: CQ Minimisation

Definition 10.5: A conjunctive query Q is minimal if:

- for all subqueries Q' of Q (that is, queries Q' that are obtained by dropping one or more atoms from Q),
- we find that $Q' \not\equiv Q$.

A minimal CQ is also called a core.

It is useful to minimise CQs to avoid unnecessary joins in query answering.

CQ Minimisation the Direct Way

A simple idea for minimising Q:

- Consider each atom of Q, one after the other
- Check if the subquery obtained by dropping this atom is contained in Q
  (Observe that the subquery always contains the original query.)
- If yes, delete the atom; continue with the next atom

Example 10.6: Example query $Q[v, w]$:

$$\exists x, y, z.R(a, y) \wedge R(x, y) \wedge S(y, y) \wedge S(y, z) \wedge S(z, y) \wedge T(y, v) \wedge T(y, w)$$

$\leadsto$ Simpler notation: write as set and mark answer variables

$$\{R(a, y), R(x, y), S(y, y), S(y, z), S(z, y), T(y, \bar{v}), T(y, \bar{w})\}$$

CQ Minimisation Example

$$\{R(a, y), R(x, y), S(y, y), S(y, z), S(z, y), T(y, \bar{v}), T(y, \bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a, y)$	$R(a, y)$	Keep (cannot map constant a)
~~$R(x, y)$~~	~~$R(x, y)$~~	Drop; map $R(x, y)$ to $R(a, y)$
$S(y, y)$	$S(y, y)$	Keep (no other atom of form $S(t, t)$)
~~$S(y, z)$~~	~~$S(y, z)$~~	Drop; map $S(y, z)$ to $S(y, y)$
~~$S(z, y)$~~	~~$S(z, y)$~~	Drop; map $S(z, y)$ to $S(y, y)$
$T(y, \bar{v})$	$T(y, \bar{v})$	Keep (cannot map answer variable)
$T(y, \bar{w})$	$T(y, \bar{w})$	Keep (cannot map answer variable)

Core: $\exists y.R(a, y) \wedge S(y, y) \wedge T(y, v) \wedge T(y, w)$

CQ Minimisation

Does this algorithm work?

- Is the result minimal?
  Or could it be that some atom that was kept can be dropped later, after some other atoms were dropped?
- Is the result unique?
  Or does the order in which we consider the atoms matter?

Theorem 10.7: The CQ minimisation algorithm always produces a core, and this result is unique up to query isomorphisms (bijective renaming of non-result variables).

Proof: exercise

How hard is CQ Minimisation?

Even when considering single atoms, the homomorphism question is NP-hard:

Theorem 10.8: Given a conjunctive query Q with an atom A, it is NP-complete to decide if there is a homomorphism from Q to $Q \setminus \{A\}$.

Proof: We reduce 3-colourability of connected graphs to this special kind of homomorphism problem. (If a graph consists of several connected components, then 3-colourability can be solved independently for each, hence 3-colourability is NP-hard when considering only connected graphs.)

Let G be a connected, undirected graph. Let $\prec$ be an arbitrary total order on G's vertices.
Query Q is defined as follows:

- Q contains atoms $R(r, g)$, $R(g, r)$, $R(r, b)$, $R(b, r)$, $R(g, b)$, and $R(b, g)$ (the colouring template)
- For every undirected edge $\{e, f\}$ in G with $e \prec f$, Q contains an atom $R(e, f)$
- For a single (arbitrarily chosen) edge $\{e, f\}$ in G with $e \prec f$, Q contains an atom $A = R(f, e)$

Claim: G is 3-colourable if and only if there is a homomorphism $Q \to Q \setminus \{A\}$

Proof

Even when considering single atoms, the homomorphism question is NP-hard:

> **Theorem 10.8:** Given a conjunctive query Q with an atom A, it is NP-complete to decide if there is a homomorphism from Q to $Q \setminus \{A\}$.

Proof (continued): ($\Rightarrow$) If G is 3-colourable then there is a homomorphism $Q \to Q \setminus \{A\}$.

- Then there is a homomorphism μ from G to the colouring template
- We can extend μ to the colouring template (mapping each colour to itself)
- Then μ is a homomorphism $Q \to Q \setminus \{A\}$

($\Leftarrow$) If there is a homomorphism $Q \to Q \setminus \{A\}$ then G is 3-colourable.

- Let μ be such a homomorphism, and let $A = R(f, e)$.
- Since $Q \setminus \{A\}$ contains the pattern $R(s, t), R(t, s)$ only in the colouring template, $\mu(e) \in \{r, g, b\}$ and $\mu(f) \in \{r, g, b\}$.
- Since the colouring template is not connected to other atoms of Q, μ must therefore map all elements of Q to the colouring template.
- Hence, μ induces a 3-colouring.

CQ Minimisation: Complexity

Even when considering single atoms, the homomorphism question is NP-hard:

Theorem 10.8: Given a conjunctive query Q with an atom A, it is NP-complete to decide if there is a homomorphism from Q to $Q \setminus \{A\}$.

Proof (summary): For an arbitrary connected graph G, we constructed a query Q with atom A, such that

- G is 3-colourable if and only if
- there is a homomorphism $Q \to Q \setminus \{A\}$.

Since the former problem is NP-hard, so is the latter.
Inclusion in NP is obvious (just guess the homomorphism). □

Checking minimality is the dual problem, hence:

Theorem 10.9: Deciding if a conjunctive query Q is minimal (that is: a core) is coNP-complete.

However, the size of queries is usually small enough for minimisation to be feasible.

Summary and Outlook

Perfect query optimisation is possible for conjunctive queries
$\leadsto$ Homomorphism problem, similar to query answering
$\leadsto$ NP-complete

Using this, conjunctive queries can effectively be minimised

Coming up next:

- How to study expressivity of queries
- The limits of FO queries
- Datalog